Visible-Surface Detection

or

Hidden-Surface Removal
Classification

- Image-space approach
  visibility at pixels
  integer precision

- Object-space approach
  exact visibility
  real precision
Algorithms

1. Z-buffering
2. Painter's algorithm
3. Scan-line algorithm
4. Area subdivision
5. Ray Casting
Z-buffering

Clip → Project → Scan Convert
→ Hidden Surface Removal

if \((p.z < i[p.x][p.y].z)\) {
    i[p.x][p.y].z = p.z;
    i[p.x][p.y].c = p.c;
}
Advantages / Disadvantages

- Simple
- Therefore hardware
- Diversity of primitives

- Memory
- Point sampling may cause aliasing
- Too much work?
Painter's algorithm

- Surfaces are sorted in order of decreasing depth
- Surfaces are scan-converted, starting at the back
Feasibility

Can a set of polygons be sorted in z and drawn in back to front order?
Painter's algorithm

1. Sort according to z.
2. Compare the depth of the back surface, S, to all S'.
   Scan convert S if
   1. No depth overlap.
   2. No overlap of bounding rectangles in the xy plane.
   3. S completely behind S'.
   4. S' completely in front S.
   5. Projections do not overlap.
else
   interchange.
   avoid infinite loops!
No depth overlap
Other cases

No xy overlap

S is completely behind S’

S’ is completely in front of S
Avoid infinite loops
BSP tree

Let P be a polygon and let Lp be the plane P resides on

The polygons can be partitioned into 3 classes:
- polygons in front of Lp
- polygons behind Lp
- polygons intersecting Lp
BSP tree
BSP-tree construction

BSPConstruct(P) {
    Choose a polygon p from P.
    Update P.
    for each polygon q in P
        if (q in front of Lp)
            Add q to Pf.
        else if (q behind Lp)
            Add q to Pb.
        else
            Split q by Lp into qb and qf.
            Add qb to Pb;
            Add qf to Pf.
    return makeTree (BSPConstruct(Pb), BSPConstruct(Pf), p);
}
Scan-line algorithm

- Simultaneously scan convert all surfaces
- At each scan line, solve 2D hidden edge problem
- Output visible spans
Scan-line algorithm
Area subdivision

Successively divide the viewing area into smaller rectangles until each rectangle is the projection of part of a single surface or no surface
Possible relationships surface/area

1. Surrounding

2. Overlapping

3. Inside

4. Outside
No further subdivisions when:

1. No polygons in the area: 
   *Fill with background color*

2. Only one polygon in the area: 
   *Fill with background color, then draw polygon*

3. Surrounding polygon in front: 
   *Fill area with its color*

4. Area <= Pixel size: 
   *Color of front polygon*
Ray Casting Method
Ray Casting Method

for each pixel (x, y){
    vz = inf;
    vo = nil;
    for each "potentially visible" object o
        if (intersect(o, x, y, &z)){
            if (z < vz)
                vo = o;
                vz = z;
        }
}